2.7 COMM DEVICES

The Communications Receiver and Communications Transmitter are two of the eight system types in Suppressor. In order for successful communications to occur between two players:

a. each must have an operational receiver and transmitter;

b. within a player, the communications receiver and transmitter must be linked in the LINKAGES section of the TDB PLAYER-STRUCTURE; and

c. the participating receivers must be ON and placed on the same communications net in the SDB.

Much of the detail of the communication net structure and message content between players is described in Functional Element 2.0 of the C^3 Template. The following sections will describe the alternate methods used to model message transmission.

2.7.1 Functional Element Design Requirements

The design requirements for the communication device functional element are:

a. Provide a capability for the user to define communication devices that operate at any frequencies in the electromagnetic spectrum. Any type of player may have any type of communication device.

b. Assume instantaneous signal transmission, but use a message delay time as defined by the user for each message type on each network type.

c. Provide the capability for the user to place each communication device on either an explicit network or an implicit network. Communications between devices on explicit networks may be degraded by jamming and terrain masking, but communications between devices on implicit networks will not.

d. For communication devices on explicit communications networks, the model will use the standard one-way energy signal transmission equation to calculate the S/I received. The following user-defined parameters will be included:

- gain of the receiver type
- internal receiver type losses
- receiver frequency
- receiver type noise
- transmitter type power
- transmitter type gain

e. Provide the capability for the user to define an antenna gain table for each type of communication transmitter and each type of receiver. The table will provide antenna gain values (in dB) for frequency, azimuth, and elevation intervals arbitrarily defined by the user (frequency is optional). The intervals will be discrete and there will be no interpolation of gain between intervals. The user may also define the vertical offset of each antenna from its platform.

f. The frequency of each specific communication device will be set to the frequency of its network at the time of use.

g. Provide the capability for the user to set the original operational status of each communication device and to point each one at a specific platform or a specific geometric location, or in a specific direction.

2.7.2 Functional Element Design Approach

Communication devices are defined in the TDB as system categories embedded in a player-structure (see Section 2.1). The system categories for communication devices are COMM-XMTR and COMM-RCVR. A communication transmitter and a receiver must both be defined if either are desired for a particular type of player.

The definition of the physical and performance characteristics of communications devices is provided in the TDB. A capability is associated with each device. Each capability consists of a set of data items which define the communications device to be modeled.

A net for each specific communications device is chosen in the SDB. The user also defines a frequency and (possibly) alternate frequencies for each network. Alternate frequencies may be defined to allow a player whose communications are being jammed to switch to another frequency.

Communications devices are defined as systems within an element. For example:

```
ELEMENT: 11     cmd_post_ele       DISCRETE QUANTITY: 1
      COMM-RCVR 112 comm_rcvr ON FREQ: 2.3 (GHZ) NET: 11 broadcast
END ELEMENT
```

During a model run, an event may be scheduled to initiate an attempt to communicate. The delay time is added to the current game time to determine the anticipated time of receipt. If communications are being modeled implicitly, no signal calculations are done and the transmission is assumed to be successful.

If communications are being modeled explicitly, the signal level is calculated at the receiver. This is done by checking to see if a signal level has already been calculated for the current transmitter/receiver pair or if they are in relative motion, which would change the signal level. Once the positions have been updated, the relative antenna orientations are determined and the range and gain are calculated. If the signal level is greater than zero, jamming interactions (if modeled) are calculated.

Next, the signal level is compared to the threshold for the receiver. If the threshold is exceeded, the message is received, and an event is scheduled for the recipient to notice it. If the message is not received, the transmitter will try an alternate frequency, if available.

2.7.3 Functional Element Software Design

The communications FE is implemented in three primary modules: YAKKER, YAKSIG, and YAKNEX. YAKKER is the top-level module that schedules the communication event between two players. YAKSIG is called for explicit networks to compute signal-to-

interference ratios, and YAKNEX is called to get the next communication event on the pending queue.

The top-level design for subroutine YAKKER is:

```
*begin logic to control communications event:
   *initialize receipt time to game time;
   *look up pointer to message recipient and message type;
   *look up pointer to net coordination header;
   *if(a message is to be sent)then;
      *invoke logic to get reception time from time delay;
      *when recipient is not dead:
         *when explicit communications:
            *invoke logic to find receiver location;
            *invoke logic to calculate signal level at receiver;
            *when any signal strength:
               *calculate commo receiver noise level;
               *invoke logic to calculate jamming noise level;
               *set quality to recognized if threshold exceeded;
               *when operator thinks the commo is being jammed:
                  *set operator thinks the commo is being jammed;
               *otherwise operator thinks the commo is not jammed:
                  *set operator thinks commo is not jammed;
               *end of test if operator thinks commo is jammed.
            *end of test for any signal strength.
            *store flag and time the operator thinks the commo is
            * being jammed;
         *end of test for implicit or explicit comm.
         *allocate commo results header;
         *write "is transmitting message" incident;
         *when signal quality was not high enough:
            *schedule sender to notice this;
            *when operator thinks the commo is being jammed &
            * there are alt. frequencies:
               *get next available frequency;
               *write out "starts to change freq" message;
               *invoke logic to adjust jammer spots focused at rx;
            *end of test if operator thinks the commo is being
            * jammed & alt. frequencies.
         *otherwise, possibly reset counters due to successful msg
            *loop, for each specified assign command chain:
               *when recipient is commander on this chain:
                  *reset comm. loss times and counters;
               *end of test if recipient is commander on chain.
            *end of loop for each specified assign command chain.
         *end of test for minimum signal quality.
         *schedule player to notice results;
      *end of test for recipient is dead.
      *decrement waiting time by time delay;
      *set the status of transmitter flag to transmitting;
   *end of test for message sent.
   *try to send next message;
   *when end of last transmission:
      *set status in coordination header to not busy;
      *set the status of transmitter flag to not transmitting;
   *otherwise, message has been sent:
      *when beginning of last transmission:
         *invoke logic to schedule end of transmit event;
```

```
         *end of test for beginning of last.
      *end of test for end of last.
      *recycle message event if end of transmit;
   *end of logic for YAKKER.
```

The top-level design for subroutine YAKSIG is:

```
   *begin logic to provide signal level at receiver terminal:
      *look up pointer to commo receiver data;
      *search for net buffer for sender;
      *when sender/receiver pair not on list:
         *add buffer for sender/receiver pair;
      *end of test for sender/receiver pair.
      *when signal level has not been stored or either is moving:
         *invoke logic to update sender location;
         *look up pointer to commo transmitter data;
         *when edge pointers are null and either moving:
            *invoke logic to check for line of sight;
         *end of test for edge pointers and either moving.
         *when line of sight exists:
            *invoke logic to calculate gain and range;
            *look up transmitted power;
            *account for transmission losses;
            *solve radio equation for received power;
         *end of test for line of sight.
      *end of test for signal level stored.
   *end of logic for YAKSIG.
```

The top-level design for subroutine YAKNEX is:

```
   *begin logic to determine next talking event for a net:
      *loop, while waiting list entries and message not sent:
         *reset pointers to last block and top list entry;
         *when message header exists:
            *when sender is not dead:
               *invoke logic to schedule message transmission;
            *end of test for sender is not dead.
            *when message buffer exists:
               *delete entry from list;
            *end of test for message buffer.
         *end of test for message header exists.
         *recycle waiting list entry;
      *end of loop for waiting list entries.
   *end of logic for YAKNEX.
```

2.7.4 Assumptions and Limitations

Message quality is not incrementally degraded. It is assumed to be perfectly received or not received depending on the signal-to-interference ratio.

2.7.5 Known Problems or Anomalies

None.